Digitizing COVID-19 Policy Documents and Measuring Plain Text Fidelity

Benjamin J. Radford[1,2,3], Jingjing Gao[1,4], Iván Flores Martínez[1,4], and Jason Windett[1,5]

[1]*University of North Carolina at Charlotte, Charlotte, NC 28223, USA.*
[2]*Email: benjamin.radford@uncc.edu*
[3]*Assistant Professor of Public Policy, Assistant Professor of Political Science and Public Administration*
[4]*Graduate Student, Public Policy*
[5]*Associate Professor of Public Policy, Associate Professor of Political Science and Public Administration*

Abstract

The COVID-19 pandemic led to a flurry of executive orders (EO) and policy directives at the state and local level as the United States struggled to adapt policy to mitigate the public health crisis. It also resulted in an explosion of research on both the coronavirus itself and the pandemic's societal impact. Unfortunately, many of those policies were written, signed, scanned, and uploaded in portable document format (PDF) to government websites. This resulted in their contents being digitized as images rather than as machine-readable text. To facilitate current research demands, we have used optical character recognition (OCR) tools to extract machine-readable text from these documents and made these "plain text" versions available online. Here we describe the pre and postprocessing steps as well as provide an evaluation of the resulting document quality. We suggest unsupervised methods for scoring output texts that can be applied to other optical character recognition tasks when ground truth plain texts are unavailable. We show that simple preprocessing modestly improves OCR performance on scanned orders and directives.

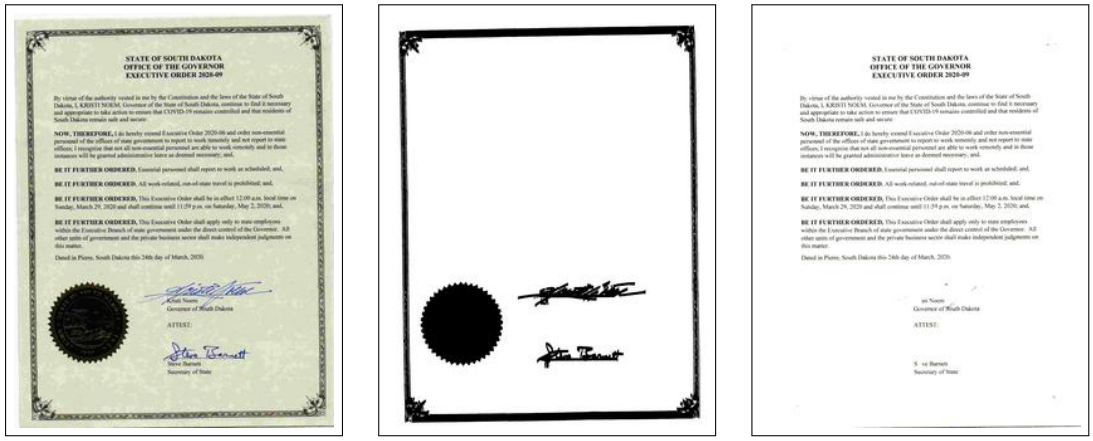
Keywords: Optical character recognition, executive orders, COVID-19.

1 Introduction

The COVID-19 pandemic has prompted researchers from many disciplines to turn their attention to the novel coronavirus. The rapidly evolving government policies are of particular interest to social scientists. As of late May, 2020, we have identified 2,213 executive orders (EO), policy directives, and declarations addressing COVID-19 from US states and counties. Some states have issued multiple policy documents per day during the crisis. Unfortunately, the texts of many of these orders are unavailable in machine-readable form. Unlike the federal government which is required to make all textual data as machine readable, states and other governments may archive in any format. As such, most of the physical documents themselves are signed and scanned as images, then compiled into PDF (portable document format) files.[1] In order for researchers to better make use of the contents of these orders, we have applied optical character recognition (OCR) techniques to extract the machine-readable "plain texts" from these documents. Plain text, or machine-readable, here refers to digital representations of alphanumeric characters, symbols, and whitespace representing human-readable content and lacking any markup, styling, or formatting.

We are making both the original, unprocessed, documents and their plain text counterparts available to the public at URLremovedforpeerreviewanonymity. Below, we describe the process we used to produce plain text versions of these documents and evaluate the quality of the resulting data.

1. A small number of the PDFs are machine-readable. However, for consistency, we treat all PDF documents as non-machine-readable images.



(a) Original page. (b) Large connected components. (c) Page after preprocessing.

Figure 1. Example of PDF preprocessing stages on South Dakota EO 2020-09.

2 Digitizing Executive Orders

We divide our method for extracting text from executive orders into three stages: preprocessing, optical character recognition, and postprocessing.

2.1 Preprocessing

We begin by rendering the PDF documents to portable network graphics (PNG) format page-by-page. Each page image is then converted into a bitmap and operated on independently. There are two primary image-based preprocessing steps.

First, we use k-means to segment the foreground and background of the image. Every pixel in the image is defined by a red, green, blue color value (RBG). k-means with $k = 2$ is used to partition pixels into two clusters based on their color values. The larger of these corresponds to the background of the image (white or off-white in most cases) while the smaller corresponds to foreground elements. All pixels that are nearest to the majority cluster centroid, those in the "background cluster," are re-colored to white. This removes subtle background patterns in the document's stationery or other noise. Because the images are large and often comprise many million pixels, a sample of 10,000 pixels is chosen for training the k-means model. Once trained, assigning all pixels to either of the clusters is fast.

Second, we use an algorithm to detect connected components (contiguous shapes) on each page. We transform the page into grayscale and then binarize it into black and white based on a threshold.[2] To identify connected components in the foreground elements of the binarized image, we use the two-pass algorithm described by Wu, Otoo, and Suzuki (2009). Any connected components that occupy 0.25% to 99% of the pixels on the page are made white (i.e. background). This removes decorative borders, state seals, most signatures, and header decorations.

2.2 Optical Character Recognition

Tesseract is an open source OCR engine that was originally developed at Hewlett Packard Laboratories and is now sponsored by Google (Vincent 2006). Early versions of Tesseract utilized a multi-step OCR process that included contour detection, polygonal approximation, and a two-pass adaptive classifier to recognize characters (Smith 2007). We use the latest release of Tesseract that instead

2. Our images take values 0 (black) to 255 (white). All values lower than or equal to 254 are mapped to 0; values of 255 remain the same. This means all foreground pixels identified by k-means are black and all background pixels are white. We use a multi-pass convolution to dilate the foreground elements prior to identifying connected components.

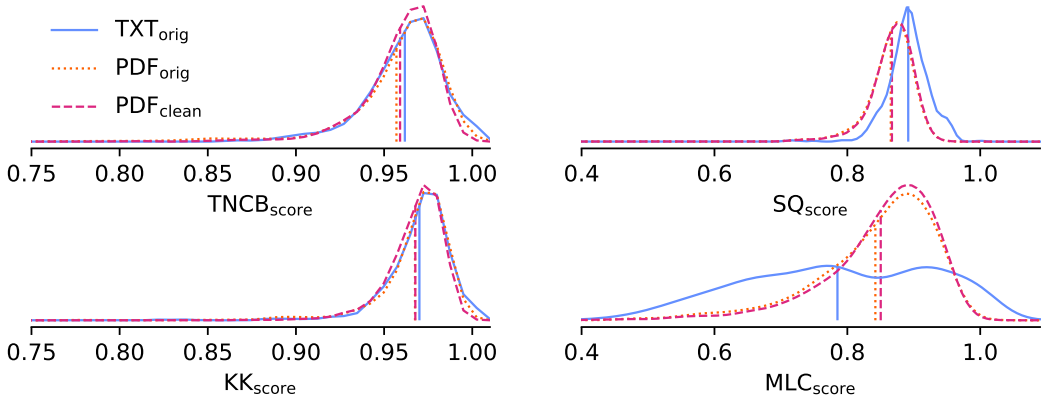


Figure 2. Distribution of text quality scores for documents in each of the three corpora. Note the differing x-axis scales. Low scores for **TXT_{orig}** on $\text{MLC}_{\text{score}}$ are due, at least in part, to single-word lines that result from menus of hyperlinks that are retained in the plain text representations of html pages.

relies on a convolutional long short term memory (hereafter LSTM) recurrent neural network.

We use the default Tesseract v.4.0 configuration with one exception. We specify page segmentation mode 4: *Assume a single column of text of variable sizes*. Tesseract must segment the text portions of an image prior to inputting those image segments into the LSTM model. Among the many options are the instructions to treat the image as a single line of text, a single word, a uniform block, or many separate blocks. The majority of documents in our corpus are single-columned, but the font sizes and alignments may change within a single page of a single document. Furthermore, while there is vertical whitespace in some documents, we found that these vertical breaks rarely corresponded to independent columns and instead corresponded to specific within-line formatting. Therefore, we selected the single column mode to prevent these vertical whitespace breaks from resulting in segmentation of the page into multiple columns.[3]

2.3 Postprocessing

We take limited postprocessing steps. Tesseract frequently fails to properly identify the character `I`, so we rely on heuristics to correct some instances of this. The characters `]`, `[`, and `|` are replaced with `I`.[4] Hyphens at the ends of lines may indicate a single word has been split at the line break or that a hyphenated word is split. When we encounter end-of-line hyphens, we remove the hyphen and concatenate the last word of the hyphenated line with the first word of the subsequent line. This candidate word is then checked against an English dictionary. If the unhyphenated candidate word is in the dictionary, it is accepted; otherwise, we remove the line break but maintain the hyphen to produce a single hyphenated word. Consecutive pages are joined by a double line break.

3 Evaluation

OCR output quality is ideally evaluated via comparison to the true text of the document in question. Unfortunately, in applied settings such as ours, the original machine-readable texts of the executive orders are unavailable. Were this not the case, we would have simply relied on the machine-readable texts all along and not bothered with OCR. We refer to the evaluation of OCR processes for which the ground truth texts are unavailable as "unsupervised evaluation." Here, we recommend a strategy for unsupervised evaluation of applied OCR tasks. In particular, we report four unsupervised OCR scores

3. For example, in its default configuration, Tesseract would sometimes interpret a series of aligned whereas clauses as two columns: one consisting of a series of "whereas" and another consisting of the clauses themselves.

4. `]` and `[` are only replaced when they do not appear as a pair within a single line.

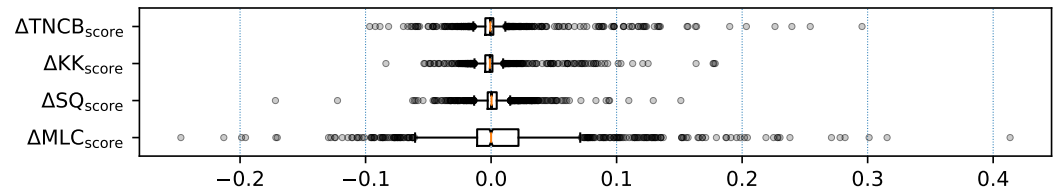


Figure 3. Difference in scores per document, $\mathbf{PDF_{clean}} - \mathbf{PDF_{orig}}$.

on three different document sets: the extracted texts from PDFs that have not been preprocessed ($\mathbf{PDF_{orig}}$), the extracted texts from the same PDFs after they have been preprocessed ($\mathbf{PDF_{clean}}$), and the true text from COVID-19 policy orders that were originally distributed in machine-readable format ($\mathbf{TXT_{orig}}$).[5]

The first two scoring methods, TNCB_{score} and KK_{score}, are heuristic approaches that identify "garbage" tokens (whitespace-delimited groups of symbols) and valid word tokens (Taghva *et al.* 2001; Kulp and Kontostathis 2007).[6] The third method for identifying garbage tokens is to match tokens against a known dictionary.[7] Tokens that are not included in the dictionary and are not Roman or Arabic numerals are considered garbage tokens. Because our documents are written primarily in English and, in some cases, Spanish, we accept either English or Spanish words. We also add "COVID-19", "COVID19", and "coronavirus" to the dictionary. This measure is called the "simply quality" score, SQ_{score} (Alex and Burns 2014). For all three measures, the score we report on a per-document basis is $1 - (\text{\# garbage tokens in document}/\text{\# tokens in document})$. Therefore TNCB_{score}, KK_{score}, and SQ_{score} can be interpreted as the proportion of valid word tokens present in a document.[8] Alex and Burns (2014) recommend a document-level threshold of $\text{SQ}_{score} \geq 0.7$ for data mining tasks based on a comparison with human-coded quality scores.

We also compute 1 − (mean language uncertainty), a measure proposed by Baumann 2015. Using a Naive Bayes classifier trained on character n-grams to estimate the language of a given piece of text, we compute the probability that each given line in a document is either English or Spanish (Shuyo 2010). To match the orientation of our other scores, we report *mean language certainty*, $\text{MLC}_{score} = (1/n) \sum_{\text{lines}} (\Pr(\text{English}) + \Pr(\text{Spanish}))$, where n is the number of lines in a document. Therefore, this measure can be interpreted as the mean confidence that a given plain text document is valid English or Spanish.

Figure 2 depicts the distribution of scores for each data set with respect to the four measures. While $\mathbf{PDF_{clean}}$ scores lower than the $\mathbf{TXT_{orig}}$ on three of four metrics, it does so by only a few percentage points on average. Furthermore, the vast majority of documents surpass the 0.7 threshold for SQ_{score}. The close overlap in score distributions between the original plain text data set and the text extracted via OCR gives us confidence that the OCR process has produced data suitable for further research efforts.

The pre and postprocessing steps make very little difference, on average; the majority of OCR documents score almost identically whether or not they are pre and postprocessed. However, the positive skews apparent in Figure 3 illustrate that some documents see their scores improve by over 30 percentage points due to pre and postprocessing. Paired t-tests indicate that pre and postprocessing result in significant (but not necessarily substantial) mean gains across three of

5. $\mathbf{TXT_{orig}}$ primarily comprises texts derived from `.html` documents.
6. These heuristics were originally designed for cleaning OCR output. We have adapted them for use in quality assessment.
7. We use the Hunspell spell checker: http://hunspell.github.io.
8. These are likely to be conservative measures as *many* valid tokens will be misclassified as garbage; for example: some within-document section references, email addresses, web addresses, phone numbers, brand names, and specific legal or medical terminology.

the four measures: $\Delta\text{TNCB}_{\text{score}} = +0.002$, $\Delta\text{KK}_{\text{score}} = -0.000$, $\Delta\text{SQ}_{\text{score}} = +0.002$, and $\Delta\text{MLC}_{\text{score}} = +0.008$ with p-values of 0.004, 0.996, 1.48×10^{-5}, and 5.16×10^{-11}, respectively.

4 Discussion

While we have confidence that these plain text versions of COVID-19 related policies will prove valuable to researchers, the OCR process described here is not perfect. One notable issue that occurs frequently is that the connected components removal preprocessing step removes signatures. When signatures overlap with typed text, obscured portions of the text are also removed. This results in occasional degradation of the the names, titles, and, less frequently, dates at the ends of documents. A small number of documents have manual markups that are improperly recognized. For example, sections of documents that have been manually struck through with pen will not be properly represented in plain text. Character homoglyphs will also sometimes lead to mistakes in the OCR process. 5 and S, for example, may be confused for one another. Underlining of text can cause that portion of text to be identified as a connected component above the set threshold and therefore removed. This can affect, for example, underlined hyperlinks. Speckles and decoration that are not completely removed by our preprocessing often result in erroneous output characters; these occur especially frequently at the beginnings and endings of documents.

Despite the outstanding issues briefly described above, we find that OCR technologies produce usable plain text data from government documents which are, in terms of quality, on par with native plain text documents.

References

Alex, B., and J. Burns. 2014. "Estimating and Rating the Quality of Optically Character Recognised Text." In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage,* 97–102. DATeCH '14. Madrid, Spain: Association for Computing Machinery. ISBN: 9781450325882. doi:10.1145/2595188.2595214. https://doi.org/10.1145/2595188.2595214.

Baumann, R. 2015. "Automatic evaluation of OCR quality." */etc (blog)* (March). https://ryanfb.github.io/etc/2015/03/16/automatic_evaluation_of_ocr_quality.html.

Kulp, S., and A. Kontostathis. 2007. "On retrieving legal files: Shortening documents and weeding out garbage." In *TREC,* edited by E. M. Voorhees and L. P. Buckland, vol. Special Publication 500-274. National Institute of Standards / Technology (NIST).

Shuyo, N. 2010. *Language Detection Library for Java.* http://code.google.com/p/language-detection/.

Smith, R. 2007. "An Overview of the Tesseract OCR Engine." In *Proceedings of the Ninth International Conference on Document Analysis and Recognition - Volume 02,* 629–633. ICDAR '07. USA: IEEE Computer Society. ISBN: 0769528228.

Taghva, K., T. Nartker, A. Condit, and J. Borsack. 2001. "Automatic Removal of "Garbage Strings" in OCR Text: An Implementation." *Proceedings of the 5th World Multi-Conference on Systemics, Cybernetics and Informatics.*

Vincent, L. 2006. "Announcing Tesseract OCR." *Google Code Blog* (August). http://googlecode.blogspot.com/2006/08/announcing-tesseract-ocr.html.

Wu, K., E. Otoo, and K. Suzuki. 2009. "Optimizing Two-Pass Connected-Component Labeling Algorithms." *Pattern Anal. Appl.* (Berlin, Heidelberg) 12, no. 2 (February): 117–135. ISSN: 1433-7541. doi:10.1007/s10044-008-0109-y. https://doi.org/10.1007/s10044-008-0109-y.